Adaptive Resource Allocation for Virtualized Base Stations in O-RAN With Online Learning

Kalntis, Michail; Iosifidis, George; Kuipers, Fernando A.

DOI
10.1109/TCOMM.2024.3461569

Publication date
2025

Document Version
Final published version

Published in
IEEE Transactions on Communications

Adaptive Resource Allocation for Virtualized Base Stations in O-RAN With Online Learning

Michail Kalntis, *Graduate Student Member, IEEE*, George Iosifidis, *Member, IEEE*, and Fernando A. Kuipers, *Senior Member, IEEE*

***Abstract*— Open RAN systems, with their virtualized base stations (vBSs), offer increased flexibility and reduced costs, vendor diversity, and interoperability. However, optimizing the allocation of radio resources in such systems raises new challenges due to the volatile vBSs operation, and the dynamic network conditions and user demands they are called to support. Leveraging the novel O-RAN multi-tier control architecture, we propose a new set of resource allocation *threshold policies* with the aim of balancing the vBSs' performance and energy consumption in a robust and provably optimal fashion. To that end, we introduce an online learning algorithm that operates under minimal assumptions and without requiring knowledge of the environment, hence being suitable even for "challenging" environments with non-stationary or adversarial demands and conditions. We also develop a meta-learning scheme that utilizes other available algorithmic schemes, e.g., tailored for more "easy" environments, by choosing dynamically the best-performing algorithm; thus enhancing the system's effectiveness. We prove that the proposed solutions achieve sub-linear regret (zero optimality gap), and characterize their dependence on the main system parameters. The performance of the algorithms is evaluated with real-world data from a testbed, in stationary and adversarial conditions, indicating energy savings of up to $64.5\,\%$ compared with several state-of-the-art benchmarks.**

***Index Terms*— O-RAN, online learning, bandit feedback, network optimization, virtualization, expert advice.**

I. Introduction

A. Motivation & Background

THE importance of base station virtualization is best illustrated by the flurry of industrial and academic activities that focus on the development of virtualized and Open Radio Access Network (O-RAN) architectures [1]. The O-RAN Alliance, for example, is a global initiative aiming to softwarize and standardize RANs so as to improve their performance, reduce their costs, and lower the entry barrier towards a wider vendor ecosystem. At the core of this transformation are the virtualized Base Stations (vBSs), such as srsLTE [2] and OpenAirInterface (OAI) [3], which offer OPEX/CAPEX savings and performance gains, since their operational parameters can be adjusted with high granularity at runtime [4]. Alas, these benefits come at a cost. Softwarized base stations are found to have less predictable performance and more volatile energy consumption [5], [6], [7], an issue that is amplified when instantiating them in general-purpose computing infrastructure. This induces operation and cost uncertainties at times when there is an increased need for robustness and performance guarantees in mobile networks. Therefore, it becomes imperative to understand how to configure or schedule these vBSs (i.e., how to allocate their resources) without relying on strong assumptions or compromising network performance, in order to unblock their deployment and maintain energy costs at sustainable levels.

The O-RAN architecture offers new opportunities to achieve this goal. Namely, the emerging O-RAN standards [8], [9] have provisions for multi-tier control solutions for resource management that can be implemented centrally, i.e., by the RAN Intelligent Controller (RIC), and enforced at different time-scales. In particular, our focus here is on non-Real-Time (non-RT) policies that determine the operation envelope (or resource allocation *thresholds*) of the vBSs over time intervals (rounds) of a few seconds. These policies are fed to, and enforced by, the real-time radio scheduler of each vBS, which devises their assignments subject to global rules about, e.g., the maximum transmission power, the eligible modulation and coding schemes (MCS), and so on. Such centralized threshold policies have been recently introduced, e.g., see [7], [10], and [11], and have several practical advantages. First, O-RAN includes heterogeneous base stations that are challenging, if not impossible, to configure directly by intervening with their real-time schedulers. The global non-RT policies, on the other hand, offer an easy path to shape the operation of each vBS. Secondly, using such central policies, the O-RAN controllers can coordinate the operation of their vBSs in a unified fashion, managing jointly their resources, and also use AI/ML mechanisms that can benefit from this centralized view.

Nevertheless, the effective design of such policies is a new and particularly intricate problem. Due to their coarse time scale (seconds) and unlike the typical Radio Resource Management (RRM) decisions (updated in msecs), these policies do not have access to the network conditions and user traffic that will be realized during the interval they will be

Received 15 June 2023; revised 6 February 2024 and 22 August 2024; accepted 4 September 2024. Date of publication 16 September 2024; date of current version 19 March 2025. This work has been supported by the European Commission through Grant No. 101139270 (ORIGAMI) and the National Growth Fund through the Dutch 6G flagship project "Future Network Services". The associate editor coordinating the review of this article and approving it for publication was Z. Chen. *(Corresponding author: Michail Kalntis.)*

The authors are with the Department of Software Technology, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: m.kalntis@tudelft.nl; g.iosifidis@tudelft.nl; f.a.kuipers@tudelft.nl).

Digital Object Identifier 10.1109/TCOMM.2024.3461569

applied. And, further, these parameters can change arbitrarily during such large time windows, not necessarily following a stationary distribution. Moreover, due to the heterogeneity and volatile operation of the vBSs, the effect of such policies on the KPIs of interest is challenging, if not impossible, to predict or quantify with analytical expressions. Coupled with the typically large number of possible policies, this compounds finding the optimal policy for each vBS. In light of these observations, it is not surprising that the first works in this area focused on O-RAN operations under static network conditions and demands [7], [11].

Our work addresses the following question: *how to design robust vBS non-RT policies that offer performance/cost guarantees without relying on strong assumptions and avoid sub-optimal operation points?* We consider O-RAN policies that determine thresholds (upper bounds) for key vBS operation knobs, namely for the vBS transmission power, the eligible MCS, and the Physical Resource Blocks (PRB), in the Uplink (UL) and Downlink (DL). Each policy is updated at a non-RT scale, based on the performance, cost, and context (conditions and demands) observations of the past, and is subsequently fed to real-time schedulers that assign the vBS radio resources.

B. Contributions

Our *first contribution* is the design and evaluation of a robust *adversarial* bandit algorithm, cf. [12], which: *(i)* identifies effective policies without relying on assumptions about the environment; *(ii)* offers tight performance guarantees; *(iii)* is oblivious to the (unknown and possibly time-varying) vBS performance; and *(iv)* has minimal and constant (in observations and time) memory requirements, as it uses closed-form expressions that can be calculated even in real-time and in resource-constrained platforms. The performance is quantified using a combined metric of effective throughput modulated by the traffic demands, and energy consumption, where the latter can be prioritized via a weight parameter. It is important to note that no assumption (e.g., convexity) is made on the performance function (i.e., we follow a black-box approach). For the optimality criterion, we use *regret*, where we compare the time-aggregated performance of the algorithm with that of a hypothetical benchmark that is designed with the help of an oracle providing access to all future/necessary information.

The *second contribution* is the expansion of this learning algorithm with a *meta-learning* scheme, which boosts the performance whenever possible. Namely, the robustness of the algorithm described above means it might be conservative when the environment is *easy*, e.g., when the network has access to context information, or if the channel qualities and traffic demands are stationary or exhibit periodicity [13]. For these cases, data-efficient solutions such as [7] can leverage the available information to identify optimal policies faster. Hence, the question that arises naturally is how to combine the required robustness without compromising learning performance (in terms of convergence speed) whenever the environment is easy. To address this, we introduce a *meta-learner* that selects intelligently among policies proposed by different algorithms that rely on, and perform better under, different assumptions. A key challenge is that the learning happens on two levels: the meta-learner has to learn which is the best-performing algorithm, and each algorithm has to learn which is the best-performing policy, while partial (i.e., bandit) feedback is received on both levels. Our approach addresses this challenge through a framework that guarantees the network will perform as well as the best-performing algorithm.

In summary, the main technical contributions of this paper are the following:

- We study the vBS resource allocation problem in its most general form, i.e., in non-stationary/adversarial environment and without knowledge of vBS throughput/cost functions. Our proposed scheme achieves sub-linear regret and has minimal computation and memory overhead [12]. This is the first work applying *adversarial* bandits to vBS resource allocation.
- We devise a meta-learning strategy that entails the use of algorithms tailored to different environments and obtains sub-linear regret with respect to the best algorithm, in each case.
- We use real-world traffic traces and testbed measurements to demonstrate the weaknesses of prior works [7], as well as the efficacy of the proposed learning algorithm in a battery of representative scenarios. We release the source code[1] of our implementation online, along with detailed documentation.

C. Organization

Sec. II discusses the related work, and Sec. III introduces the O-RAN background and the system model. In Sec. IV, we present the main learning algorithm, and in Sec. V, the meta-learner. Sec. VI illustrates the performance evaluation and we conclude in Sec. VII.

II. Related Work

Resource management for *softwarized* networks can be broadly classified into models that relate policies to performance functions, model-free approaches, and Reinforcement Learning (RL) techniques. Model-based examples include [5] and [14], which maximize the served traffic subject to vBS computing capacity, but do not capture the impact that the hosting platform, the environment, or user demands may have on the vBS's operation [6]. Model-free approaches employ Neural Networks to approximate the performance functions of interest [15], yet, their efficacy depends on the availability of representative training data. Finally, RL solutions [16] use runtime observations and have been used, for example, in interference management [17], vBS function splitting [18], and handover optimization [19]. The disadvantages of all these works are the curse of dimensionality and the lack of robust convergence guarantees [20]. Following an akin approach, contextual bandit algorithms are employed to optimize video streaming rates [21] or handover decisions [22]; assign CPU time to virtualized BSs [10], and control millimeter-Wave networks [23]. Unfortunately, these works require access to

[1] https://github.com/MikeKalnt/MetBS

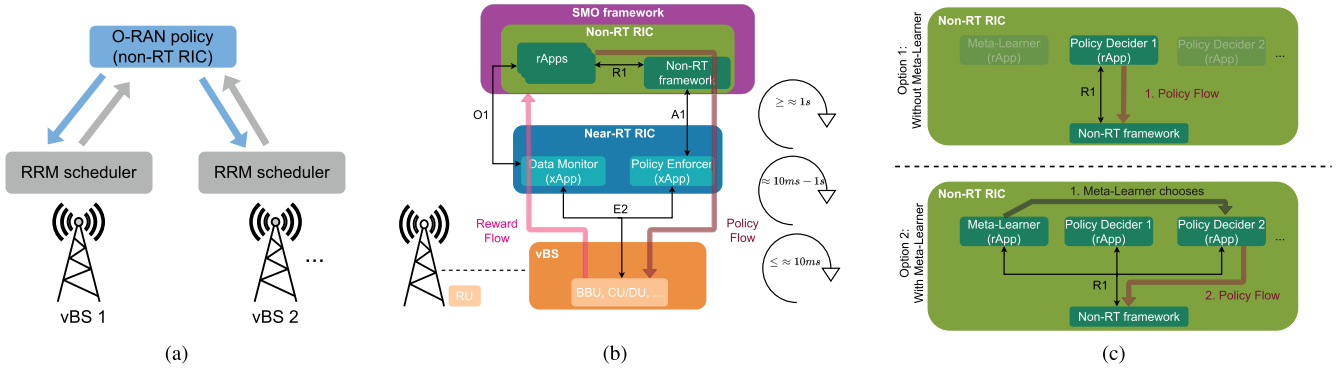

Fig. 1. O-RAN-compliant architecture & policy workflow. **(a)**. The proposed policy operates in the non-RT RIC and decides MCS, Power and PRB thresholds that are sent to each vBS's scheduler. **(b)** The key building block is the Non-RT RIC, hosted by the Service Management and Orchestration (SMO) framework, and the Near-RT RIC. The system has three control loops: (*i*) Non-RT, which involves large-timescale operations with execution time $\geq$1 s, (*ii*) Near-RT (>10 ms), and (*iii*) RT ($\leq$10 ms). **(b)** Policy Flow for the Non-RT RIC with (bottom) and without (top) an rApp implementing a meta-learner.

context information. More recently, Bayesian learning has been used for RRM, see [24] and references therein, but these solutions also need access to context information and converge only under stationary conditions.

We take here a different approach, based on *adversarial* bandits, cf. [25], which is robust to adversarial or non-stationary contexts (channel qualities and traffic demands), and has low memory and computation requirements. This latter feature is in stark contrast to RL (with sizeable memory space required to store all space-actions combinations) and Bayesian techniques [7], [24] which involve slow matrix inversions [26]. Such adversarial/non-stationary environments are increasingly common due to highly volatile network conditions [27] and traffic demands [28]. Furthermore, we draw ideas from the expert-learning paradigm [29] and enrich our policy decisions with a meta-learning scheme that combines our adversarial learning algorithm (that can be at times conservative) with any other algorithm (e.g., [7]) that performs better on more *easy* scenarios where the environment is predictable (e.g., when stationary). This meta-algorithm obtains the best of both approaches, and succeeds in being both fast-learning and robust; an idea that has been used in online learning [30], but not in network management.

We employ the above method to tackle a joint performance and energy cost optimization problem. Similar (in scope) formulations have been extensively studied in the literature. For instance, [31] considered a joint user association, spectrum, and power allocation model for throughput optimization; [32] focused on spectrum and energy efficient beamforming; and [33] optimized the spectrum and power assignment using genetic algorithms. Nevertheless, such approaches assume the system and user-related parameters to be fixed and known. On the other hand, many dynamic formulations rely on RL to optimize energy and performance, e.g., [34]; or on variants of the seminal CGP-UCB algorithm [7]. The main limitation of these works is the need to know contextual information (channels, user demands, etc.), and the lack of optimality guarantees, mainly in non-stationary conditions. Our approach is instead tailored to handle the inherent performance and cost volatility of O-RAN systems without access to context and provides optimality guarantees against competitive oracles.

Finally, a key difference between our work and the above RRM literature lies in our emphasis on non-RT RAN policies. These policies serve as operational thresholds for the real-time vBS (i.e., RRM) schedulers and are facilitated by the O-RAN architecture, which has provisions for such tiered control loops [8], [9]. This approach enables centralized management of multiple BSs without disrupting their RRM functionality. Recent works [35], [36] use RL for selecting the slicing and scheduling policies in O-RAN (i.e., RRM schedulers, see Sec. III). Nevertheless, our policy thresholds operate on a higher timescale (i.e., Non-RT) and are fed to these vBS schedulers, which make RRM decisions subject to our provided thresholds; also, they learn in an online (not offline) manner and adapt to environment changes, even if these changes happen drastically.

A recent stream of works has followed this path to design such non-RT operation thresholds. Namely, [7] uses CGP-UCB to identify thresholds for the maximum allowed transmission power and MCS to reduce the energy consumption of base stations; [11] follows a similar approach but focuses on different performance KPIs; and [10] decides maximum MCS and duty cycles through deep learning. Unlike these works, our solution is the first to provide optimality guarantees for non-stationary environments and without requiring access to context, while through the proposed meta-learner we can combine and benefit from other algorithms (e.g., [7]) when they perform well.

III. Background and System Model

A. O-RAN Background

Our model follows the O-RAN architecture [1], [8], [9], which has provisions for resource management and decision-making at different time scales: at seconds/minutes level (i.e., in Non-RT RIC through rApps[2]), and the millisecond level (i.e., in Near-RT RIC through xApps). The proposed algorithms can be implemented as rApps at the Non-RT RIC,

[2]The O-RAN general terms rApp and xApp are used to describe applications in the RIC at Non-RT (for rApp) or near-RT (for xApp) time scale, which can implement various RAN control algorithms or other pertinent services; see [1].

aiming to learn energy-efficient threshold radio policies [9]. These policies are essentially *threshold rules* regarding the maximum MCS, PRB, and transmission power that each vBS, in real-time, is allowed to use. Specifically, these rules are communicated to the vBSs under the RIC, so as to guide their RRM schedulers which allocate the radio resources in real-time accordingly, see Fig. 1(a). This approach is in line with a recent stream of papers [1], [7], [10], [11] proposing threshold policies and exploits the multi-tier (multi-timescale) architecture of O-RAN to offer centralized control of multiple vBS, without intervening to their (often proprietary/hardcoded) real-time schedulers.

We consider here the typical vBS comprising a Base Band Unit (BBU) hosted by an off-the-shelf platform attached to a Radio Unit (RU).[3] This tiered control approach can be seen in Policy Flow, Fig. 1(b) and 1(c) top. At each round, with typical duration ~ 1 second, the *Policy Decider* (i.e., algorithm) devises the threshold policy which is communicated (via the A1 interface) to the Near-Real-Time (Near-RT) RIC, where an xApp (termed *Policy Enforcer*) forwards it to the different vBSs.[4] This makes a two timescale system where the policy is devised at each round (seconds) and the vBSs schedulers update their typical RRM decisions every slot (mseconds), based on these rules.

O-RAN's flexibility enables the usage of O1 to receive/forward the policy directly from/to the real-time scheduler [36]. Nevertheless, our decision to involve xApps through the Near-RT RIC stems from providing a more general framework, where, e.g., another xApp could take the thresholds we provide, save them to a database, and perform additional actions to ensure that those thresholds are respected or make any other inference. Our modular architecture is designed to be adaptable and general enough to accommodate this, and is in accordance with recent works [1], [7].

The Policy Flow changes when including a meta-learner as another rApp (Fig. 1(c) bottom), whose goal is to discern the best Policy Decider among the employed ones. This is achieved by selecting at each round one of the available Policy Deciders, which, in turn, chooses the threshold policy. At the end of each round, the Near-RT RIC's Data Monitor computes a *reward* by aggregating the performance and cost measurements (for all slots) received via the E2 and feeds them to the selected Policy Decider via the O1 interface (Reward Flow in Fig. 1(b)). The terms Policy Decider, Policy Enforcer, and Data Monitor are introduced in this work to clarify the role of each rApp/xApp, as these last terms are generic.

B. vBS Policies

We optimize the system operation over a time horizon of $t = 1, \ldots, T$ rounds. For the DL, we define the set of the maximum allowed *vBS transmission powers*, $\mathcal{P}^{\mathrm{d}} = \left\{p_i^{\mathrm{d}}, \forall i \in \{1, \ldots, P^{\mathrm{d}}\}\right\}$, the set of highest eligible *MCS*, $\mathcal{M}^{\mathrm{d}} = \left\{m_i^{\mathrm{d}}, \forall i \in \{1, \ldots, M^{\mathrm{d}}\}\right\}$, and the set of maximum *PRB ratio*, $\mathcal{B}^{\mathrm{d}} = \left\{b_i^{\mathrm{d}}, \forall i \in \{1, \ldots, B^{\mathrm{d}}\}\right\}$, where P^{d}, M^{d}, and B^{d} denote the number of possible transmission power, MCS, and PRB ratio levels in DL, respectively.[5] The PRB ratio corresponds to the portion of the available PRBs the channel supplies, e.g., $b_i^{\mathrm{d}} = 0.2$ leads to utilization of $20\,\%$ (10 out of 50 PRBs). The DL policy for round t is denoted with $x_t^{\mathrm{d}} \in \mathcal{P}^{\mathrm{d}} \times \mathcal{M}^{\mathrm{d}} \times \mathcal{B}^{\mathrm{d}}$. Similarly, for the UL we introduce the sets $\mathcal{M}^{\mathrm{u}} = \left\{m_i^{\mathrm{u}}, \forall i \in \{1, \ldots, M^{\mathrm{u}}\}\right\}$ and $\mathcal{B}^{\mathrm{u}} = \left\{b_i^{\mathrm{u}}, \forall i \in \{1, \ldots, B^{\mathrm{u}}\}\right\}$, where M^{u}, B^{u} are the available MCS and PRB ratio levels in UL[6] and denote with $x_t^{\mathrm{u}} \in \mathcal{M}^{\mathrm{u}}$ the UL policy. Putting these together, the t-round threshold policy is:

$$x_t = (x_t^{\mathrm{d}}, x_t^{\mathrm{u}}) \in \mathcal{X}, \text{ where } \mathcal{X} = \mathcal{P}^{\mathrm{d}} \times \mathcal{M}^{\mathrm{d}} \times \mathcal{B}^{\mathrm{d}} \times \mathcal{M}^{\mathrm{u}} \times \mathcal{B}^{\mathrm{u}}.$$

C. Rewards & Costs

The first goal of the learner is to maximize the *effective* DL and UL throughputs, which depend on the aggregate transmitted data and the backlog in each direction. In line with prior works (see [7] and references therein), we use the *utility* function:

$$U_t(x_t) = \log\left(1 + \frac{R_t^{\mathrm{d}}(x_t^{\mathrm{d}})}{d_t^{\mathrm{d}}}\right) + \log\left(1 + \frac{R_t^{\mathrm{u}}(x_t^{\mathrm{u}})}{d_t^{\mathrm{u}}}\right), \quad (1)$$

where $d_t^{\mathrm{d}}, d_t^{\mathrm{u}} > 0$, with $U_t(x_t) = 0$ otherwise. $R_t^{\mathrm{d}}(\cdot)$ and $R_t^{\mathrm{u}}(\cdot)$ denote the DL and UL *transmitted data* during round t; and d_t^{d} and d_t^{u} are the respective backlogs, i.e., the traffic *demands* during t. The logarithmic transformation balances the system utility across each stream (i.e., DL and UL), but we note that other mappings (e.g., linear) might be used to capture the specifics of different applications. We have divided the transmitted data by the actual traffic demands in the respective stream (UL or DL), since the reward should naturally be defined w.r.t. the needs of the system. Similarly, one can readily extend the utility function to capture various QoS metrics, e.g., by measuring only the throughput above a certain threshold. We refrain from making assumptions about how $x_t^{\mathrm{u}}, x_t^{\mathrm{d}}$ affect the transmitted data, $R_t^{\mathrm{d}}, R_t^{\mathrm{u}}$; similarly, the traffic demands, $d_t^{\mathrm{d}}, d_t^{\mathrm{u}}$, are also considered unknown and can vary arbitrarily.[7] In this black-box approach, each threshold policy x_t (i.e., *bandit arm*) yields a reward, which we calculate a posteriori, and corresponds to the reward of the respective bandit arm. The goal of our algorithms is to learn progressively which bandit arm leads to the highest possible reward.

The second goal of the learner is to minimize the vBS energy costs. To that end, we introduce the time-varying *power cost* function $P_t(x_t)$, which depends on policy x_t in a possibly unknown fashion. Our decision to refrain from making assumptions about this function is rooted in the complexities involved in characterizing the power consumption and costs of such virtualized base stations [6]. Furthermore, this

[3]The BBU corresponds to a Long-Term Evolution (LTE) eNodeB (eNB) for a 4G network and to a New Radio (NR) gNodeB (gNB) for a 5G network. For the latter, gNB is disaggregated into the Distributed Unit (DU), and the Centralized Unit (CU).

[4]E2 nodes refer to O-RAN nodes O-CU, O-DU, O-RU, and O-eNB, which denote the CU, DU, RU, and eNB, respectively.

[5]The MCS values are predetermined, and similarly, one can quantize the power and PRB ratio values; see, e.g. [2].

[6]A maximum allowed UE transmission power is not defined since the users' transmission power has less impact on the vBS power than the MCS and PRBs in the UL. However, it can be included in x_t^{u} if deemed relevant for another application.

[7]Kindly refer to Sec. VI for details on their calculation during the evaluation of the proposed algorithms.

black-box approach allows us to capture a range of factors that might affect the consumed energy (e.g., retransmissions due to interference or time-varying electricity prices).

Putting these together, the *learner*'s criterion is the *reward* function $\tilde{f}_t\colon \mathcal{X} \to \mathbb{R}$ defined as:

$$\tilde{f}_t(x_t) = U_t(x_t) - \delta P_t(x_t), \tag{2}$$

where parameter $\delta > 0$ is set by the network operator to tune the relative priority of utility and energy costs. Parameter δ serves as a metric transformation, enabling a meaningful scalarization of U_t and P_t. Furthermore, we introduce, for technical reasons, the *scaled* reward function $f_t : \mathcal{X} \to [0, 1]$, since our learning algorithms (see Sec. IV and V) operate on that interval. An easy-to-implement mapping that ensures this normalization is:

$$f_t(x_t) = \big(\tilde{f}_t(x_t) - \tilde{f}_{min}\big)/\big(\tilde{f}_{max} - \tilde{f}_{min}\big). \tag{3}$$

Parameters $\tilde{f}_{min}$ and $\tilde{f}_{max}$ can be determined based on δ, the min/max value of power cost function, the min/max vBS transmission power, PRB ratio, MCS and traffic demands.

D. Environment

We refer to the "external" information, i.e., $\{c_t^{\mathrm{d}}, c_t^{\mathrm{u}}, d_t^{\mathrm{d}}, d_t^{\mathrm{u}}\}_{t=1}^T$ as *environment*, and it is responsible for shaping the function f_t. It is crucial to note that both reward components, U_t and P_t, vary with time t, an effect that is attributed to several factors. First, the traffic demands, i.e., d_t^{d} in DL and d_t^{u} in UL, change, sometimes drastically, in every round t, e.g., in small-cell networks where user churn is high, which affects U_t, see (1). The demands also impact the choice of MCS and PRB, leading to different processing times and, thus, different power costs. Second, the channel qualities (i.e., CQIs) in DL and UL, denoted as c_t^{d} and c_t^{u}, respectively, might vary (in slow, fast, or mixed timescales), and this affects the transmitted data R_t^{d} and R_t^{u} (hence, U_t changes even for fixed x_t) and the energy cost P_t (low CQI induces more BBU processing [6]).[8]

Importantly, we consider the environment to be *unknown* at the beginning of each scheduling round t. It is often challenging to predict the traffic demands, energy availability, channel qualities, wireless interference and other performance-related impairments that each vBS might encounter over the time window of several seconds that these threshold policies will be enforced. This, in turn, means that when we decide x_t in each round, we do not have access to f_t; and this is in notable contrast to the typical real-time radio management solutions that require accurate context information. Our model is hence oblivious to this information and this renders our solution applicable to a range of practical scenarios, such as those involving highly volatile environments and small cells where demands are non-stationary [28].

[8]The operation cost of the vBS hosting platform is subject to variations in external computing loads (e.g. when co-hosting other services or other vBS/DUs), changes in the monetary cost (or availability) of the energy price, and so on.

IV. Policy Learning for Adversarial Environments

A. Objectives & Approach

The goal of our rApp (see Policy Decider, Fig. 1) is to find a sequence of policies $\{x_t\}_{t=1}^T$ that induce rewards approaching the cumulative reward of the single best policy (*benchmark*). Formally, we employ the metric of *static expected regret*:

$$\mathcal{R}_T = \max_{x\in\mathcal{X}} \left\{ \sum_{t=1}^{T} f_t(x) \right\} - \mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t) \right], \tag{4}$$

where the first term is the aggregate performance of the benchmark (ideal) policy that can be selected only with knowledge of all future reward functions until T; and the second term measures the aggregate performance of the algorithm. The expectation in the second term is induced by any possible randomization in $\{f_t\}_{t=1}^T$ and in the selection of $\{x_t\}_{t=1}^T$ by the learner. Eventually, our objective is to devise a rule that decides the policies in such a way that the average regret, for any possible realization of rewards $\{f_t\}_{t=1}^T$, diminishes asymptotically to zero, i.e., $\lim_{T\to\infty} \mathcal{R}_T/T = 0$. Importantly, we wish to ensure this condition: *(i)* without knowing f_t when deciding x_t, and *(ii)* by observing only $f_t(x_t)$ when applying x_t, and not the complete function $f_t(x), \forall x \in \mathcal{X}$, as only *one* policy x_t in each round t can be deployed to the vBS.

The proposed scheme, named Bandit Scheduling for vBS (BSvBS), builds upon the *Exp3* algorithm [37], and its underlying idea is to learn the correct probability distribution y_t^{B} (B refers to BSvBS) from which we can sample x_t for each round t:

$$x_t \sim \mathbb{P}(x_t = x') = y_t^{\mathrm{B}}(x'), \forall x' \in \mathcal{X}.$$

The distributions $\{y_t^{\mathrm{B}}\}_{t=1}^T$ belong to the probability simplex:

$$\mathcal{Y}^{\mathrm{B}} = \left\{ y^{\mathrm{B}} \in [0,1]^{|\mathcal{X}|} \;\middle|\; \sum_{x\in\mathcal{X}} y^{\mathrm{B}}(x) = 1 \right\},$$

and are calculated in each round using the following explore / exploit rule:

$$y_t^{\mathrm{B}}(x) = \frac{\gamma}{|\mathcal{X}|} + (1-\gamma)\frac{w_t^{\mathrm{B}}(x)}{\sum_{x'\in\mathcal{X}} w_t^{\mathrm{B}}(x')}, \;\; \forall\, x \in \mathcal{X}. \tag{5}$$

This formula includes three components: (i) the exploration part, $1/|\mathcal{X}|$ which selects a policy randomly, (ii) the exploitation part, $w_t^{\mathrm{B}}(x)/\sum_{x'\in\mathcal{X}} w_t^{\mathrm{B}}(x')$, which chooses a threshold policy based on its performance up until $t-1$, where the weight $w_t^{\mathrm{B}}(x)$ tracks the reward of each policy $x \in \mathcal{X}$, and (iii) parameter $\gamma \in (0, 1]$, which prioritizes the former (explore) or the latter part (exploit).

For the latter, we employ the weight vector $w_t = \big(w_t(x) : x = 1, \ldots, |\mathcal{X}|\big)$ that tracks the success of each tested policy, which is updated at the end of each round using:

$$w_{t+1}^{\mathrm{B}}(x) = w_t^{\mathrm{B}}(x) \exp\left(\frac{\gamma \Phi_t^{\mathrm{B}}(x)}{|\mathcal{X}|}\right), \;\; \forall\, x \in \mathcal{X}, \tag{6}$$

which assigns a probability exponentially proportional to the cumulative reward $\Phi_t^{\mathrm{B}}(x)$, that accounts for the selection of

Algorithm 1 Bandit Scheduling for vBS (BSvBS)

Parameters: $\gamma = (0, 1]$
Initialize: at $t = 1$, $w_1^{\mathrm{B}}(x) \leftarrow 1,\ \forall x \in \mathcal{X}$
for $t = 1, 2, \ldots, T$ **do**
  Define the probability $y_t^{\mathrm{B}}(x),\ \forall x \in \mathcal{X}$ using (5).
  Sample next policy: $x_t \sim y_t^{\mathrm{B}}$.
  Receive & scale reward $f_t(x_t)$ using (2) and (3).
  Calculate weighted feedback $\Phi_t^{\mathrm{B}}(x),\ \forall x \in \mathcal{X}$ using (7).
  Update $w_t^{\mathrm{B}}(x),\ \forall x \in \mathcal{X}$ using (6).
end

each policy, namely:

$$\Phi_t^{\mathrm{B}}(x) = \begin{cases} f_t(x_t)/y_t^{\mathrm{B}}(x_t), & \text{if } x = x_t, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

By dividing each observed reward, $f_t(x_t)$ with the selection probability of the threshold-policy, $y_t^{\mathrm{B}}(x_t)$, we ensure the conditional expectation of $\Phi_t^{\mathrm{B}}(x)$ is the actual reward $f_t(x), \forall x \in \mathcal{X}$, meaning that Φ_t^{B} is an unbiased function estimator of the rewards [38]. Intuitively, this compensates the reward of thresholds that are unlikely to be chosen. The steps of the learning scheme are summarized in Algorithm 1, which takes as input γ and devises the ideal selection probability for each policy based on its expected reward.

B. Optimality Guarantees

The performance of Algorithm 1 is characterized in the following lemma, which holds for any possible sequence of functions $\{f_t\}_{t=1}^T$:

Lemma 1: Let $T > 0$ be a fixed time horizon. Set input parameter $\gamma = \min\left\{1, \sqrt{|\mathcal{X}| \ln |\mathcal{X}| / \big((e-1)T\big)}\right\}$. Then, running Algorithm 1 ensures that the expected regret is:

$$\mathcal{R}_T \leq 2\sqrt{(e-1)}\sqrt{T|\mathcal{X}| \ln |\mathcal{X}|} \tag{8}$$

Proof: The proof follows by tailoring the main result of [37], which provides an upper bound to (4), namely:

$$\mathcal{R}_T \leq (e-1)\,\gamma \max_{x \in \mathcal{X}} \left\{ \sum_{t=1}^{T} f_t(x) \right\} + \frac{|\mathcal{X}| \ln |\mathcal{X}|}{\gamma}. \tag{9}$$

The number of *bandit arms* in our case corresponds to the eligible policies; hence it is equal to $|\mathcal{X}|$. Given that: *(i)* the horizon T can be known in advance, and *(ii)* the rewards $f_t(x_t)$ for each chosen policy x_t at round t cannot be greater than 1 (due to the normalization described in Sec. III), we determine an upper bound g of $\max_{x \in \mathcal{X}} \left\{ \sum_{t=1}^{T} f_t(x) \right\}$ equal to T, i.e., $g = T$. By choosing the suggested γ, (9) leads to (8). □

C. Discussion for BSvBS

Algorithm 1, which operates with bandit feedback, is guaranteed to achieve the same performance as the (unknown) single best policy without imposing any conditions on the system operation, channel qualities, or traffic demands; see Lemma 1.

Algorithm 2 Meta-Learning for vBS (MetBS)

Parameters: $\eta = (0, 1]$
Initialize: at $t = 1$, $w_1^{\mathrm{M}}(j) \leftarrow 1$ and $h_0^{j,\mathrm{S}} \leftarrow \emptyset,\ \forall j \in \mathcal{A}$
for $t = 1, 2, \ldots, T$ **do**
  Define the probability $y_t^{\mathrm{M}}(j),\ \forall j \in \mathcal{A}$ using (10).
  Sample algorithm a^{i_t} according to: $a^{i_t} \sim y_t^{\mathrm{M}}$.
  Algorithm a^{i_t} recommends policy $x_t^{i_t}$ based on $h_t^{i_t,\mathrm{S}}$.
  Receive & scale reward $f_t(x_t^{i_t})$ using (2) and (3).
  Calculate weighted feedback $\Phi^{\mathrm{M}}(j),\ \forall j \in \mathcal{A}$ using (11).
  Update $w_t^{\mathrm{M}}(j),\ \forall j \in \mathcal{A}$ using (12).
  Sample ξ_t using (13).
  if $\xi_t = 0$ **then**
    block feedback of algorithm a^{i_t}, i.e., $h_t^{i_t,\mathrm{S}} \leftarrow h_{t-1}^{i_t,\mathrm{S}}$.
  else
    allow feedback of algorithm a^{i_t}, i.e., $h_t^{i_t,\mathrm{S}} \leftarrow h_{t-1}^{i_t,\mathrm{S}} \cup \big(x_t^{i_t}, f_t(x_t^{i_t})\big)$.
  end
end

Regarding the overheads of this algorithm, BSvBS depends on the number of policies $|\mathcal{X}|$ and the number of rounds T. Each round of the algorithm involves updating the probability distribution over the policies, see equation (5), which requires $\mathcal{O}(|\mathcal{X}|)$ time. Additionally, the algorithm updates the weights for each eligible threshold policy based on the reward, which again takes $\mathcal{O}(|\mathcal{X}|)$ time, see equations (6) and (7). Thus, for T rounds, the time complexity is generally $\mathcal{O}(T|\mathcal{X}|)$. Also, its space complexity is $\mathcal{O}(|\mathcal{X}|)$, as it needs to store only the weights and the probabilities for each policy. In other words, the algorithm is both robust and lightweight in terms of implementation, especially compared to its main competitor, BP-vRAN [7], which has $\mathcal{O}(T^3)$ time complexity and $\mathcal{O}(T^2)$ space complexity. Nevertheless, the robustness of BSvBS is achieved via a conservative approach that prevents the system from performing better when the conditions allow it. We tackle this issue in the following section.

V. Universal Policy Learning Through a Meta-Learner

A. Modeling & Challenges

The analysis in Sec. IV demonstrates the effectiveness of the proposed adversarial scheme in *all* environments, whether challenging or easy. However, in the latter case, alternative schemes that leverage the knowledge of the environment can achieve faster learning convergence [7]. *Our goal here is to devise a meta-learning scheme that leverages multiple algorithms, each tailored to a specific environment, and chooses dynamically the optimal one.* This idea is leveraged in online learning [30]; however, to the best of the author's knowledge, it is hitherto unexplored for resource allocation in RAN.

In practical terms, the implementation of such a meta-learning algorithm (Algorithm 2) can be realized in the

non-RT RIC, i.e., co-located with the Policy Deciders. Namely, we deploy A rApps, i.e., *algorithms* $a^j, j \in \mathcal{A} = \{1, \ldots, A\}$, each associated with a set of policies $\mathcal{X}^j$; and another rApp for the *meta-learner* that observes their performances over a time horizon of $t = 1, \ldots, T$ rounds via the R1 interface (see Fig. 1). At a time t, an algorithm $a^j, j \in \mathcal{A}$ takes as input the *full* history $h_t^j = \left\{\left(x_\tau^j, f_\tau(x_\tau^j)\right)\right\}_{\tau=1}^{t-1}$ of its previously proposed policies and their respective rewards, and proposes a policy $x_t^j = a^j(h_t^j)$. The objective of the meta-learner is to find the best performing algorithm $a^{i^*}, i^* \in \mathcal{A}$. The challenge lies in the fact that the algorithms are learning entities that update their proposed threshold policies based on bandit feedback, which in turn depends on whether they are selected by the meta-learner. In other words, at round t, the meta-learner chooses one algorithm $i_t \in \mathcal{A}$, denoted as a^{i_t}, which, in turn, proposes one policy $x_t^{i_t} \in \mathcal{X}^{i_t}$ that is deployed in the vBS; and thus, reward $f_t(x_t^{i_t})$ is returned,[9] cf. (2). Lastly, a^{i_t} updates its learning state by updating its history $h_t^{i_t} \leftarrow h_{t-1}^{i_t} \cup \left(x_t^{i_t}, f_t(x_t^{i_t})\right)$. All other algorithms, i.e., $\forall j \in \mathcal{A} : j \neq i_t$, observe no feedback and do not update their learning state at time t.

This downward spiral creates a challenging situation where the partial feedback reduces the learning capability of the meta-learner, which is further compounded by the limited chances of obtaining feedback for each policy. Without coordination between the meta-learner and the algorithms in the bandit setting, it is proven that the meta-learner will achieve linear regret, even if each of the algorithms obtains sub-linear regret if it were run on its own (and thus obtain feedback in every round) [39], [40]. To surmount this challenge, effective coordination between the algorithms and the meta-learner becomes essential. The approach we employ, inspired by the ideas presented in [40], aims to minimize the interaction required between the algorithms and the meta-learner. Other existing meta-algorithms such as [39] and [41] require feeding unbiased estimates of rewards to the algorithms, meaning that the meta-learner has access to the rewards of the algorithms and can modify them; an assumption that we want to drop in our setting.

In our case, the meta-learner can allow or block the chosen algorithm a^{i_t} from learning at round t by sending a corresponding bit (0 or 1). This means that each algorithm $a^j, j \in \mathcal{A}$ has access to *sparse* history $h_t^{j,\mathrm{S}} = \left\{\left(x_\tau^j, f_\tau(x_\tau^j)\right) \mid \xi_\tau = 1\right\}_{\tau=1}^{t-1}$, where ξ_τ is a Bernoulli random variable, i.e., $\xi_\tau \sim \mathcal{B}(\rho_\tau)$, defined by the meta-learner. More precisely, with probability $\rho_t \in (0, 1]$ at each round t, the meta-learner sends bit 1, allowing the chosen algorithm a^{i_t} to learn, i.e., update its history $h_t^{i_t,\mathrm{S}} \leftarrow h_{t-1}^{i_t,\mathrm{S}} \cup \left(x_t^{i_t}, f_t(x_t^{i_t})\right)$; otherwise, $h_t^{i_t,\mathrm{S}} \leftarrow h_{t-1}^{i_t,\mathrm{S}}$. Obviously, it is true that if $\rho_t = 1$, for $t = 1, \ldots, T$, then $h_t^{j,\mathrm{S}} \equiv h_t^j$. Intuitively, this prevents a situation where algorithms that initially find a good policy, but later experience a decline in performance, are continuously selected by the meta-learner over algorithms that explore more extensively in the early stages but achieve superior performance later. By choosing ρ_t accordingly in every round t (see the following analysis), all algorithms could observe feedback in an equal number of rounds (although the best-performing algorithms will be chosen more often) and thus have equal learning steps to improve their performance.

B. Objectives & Approach

Following this rationale, the second proposed scheme, named *Meta-Learning for vBS* (`MetBS`), builds upon [40]. Due to its similarity with Algorithm 1, we elaborate next only on its most crucial and distinct steps. The concept lies in learning the sequence of distributions $\{y_t^{\mathrm{M}}\}_{t=1}^T$ (M refers to MetBS), which enables the selection of an algorithm $i_t \in \mathcal{A}$, denoted as a^{i_t} at round t based on the following explore-exploit criteria with parameter $\eta \in (0, 1]$:

$$y_t^{\mathrm{M}}(j) = \frac{\eta}{A} + (1 - \eta)\frac{w_t^{\mathrm{M}}(j)}{\sum_{j' \in \mathcal{A}} w_t^{\mathrm{M}}(j')}, \quad \forall\, j \in \mathcal{A}. \tag{10}$$

Based on its history $h_t^{i_t,\mathrm{S}}$ and its internal mechanism of using it (e.g., `BSvBS` uses (5)), a^{i_t} outputs a policy $x_t^{i_t} \in \mathcal{X}^{i_t}$. The meta-learner observes only the reward $f_t(x_t^{i_t})$ that a^{i_t} produced, and thus, similarly to `BSvBS`, calculates an unbiased estimator for the rewards[10] of all the algorithms (even the unchosen ones):

$$\Phi_t^{\mathrm{M}}(j) = \begin{cases} f_t(x_t^{i_t})/y_t^{\mathrm{M}}(i_t), & \text{if } j = i_t, \\ 0, & \text{otherwise,} \end{cases}, \forall j \in \mathcal{A} \tag{11}$$

The weights, which determine the meta-learner's choices in each t, are updated according to:

$$w_{t+1}^{\mathrm{M}}(j) = w_t^{\mathrm{M}}(j) \exp\left(\frac{\eta \Phi_t^{\mathrm{M}}(j)}{A}\right), \quad \forall\, j \in \mathcal{A}. \tag{12}$$

Before `MetBS` proceeds to the next round, it has the ability to block algorithm a^{i_t} from acquiring feedback (i.e., learning) at this particular round t. Consequently, `MetBS` uses the following Bernoulli random variable to allow or block the feedback of a^{i_t}:

$$\xi_t \sim \mathcal{B}\left(\frac{\eta}{A\, y_t^{\mathrm{M}}(j)}\right), j = i_t. \tag{13}$$

More specifically, with probability $\rho_t = \eta/(A\, y_t^{\mathrm{M}}(j))$, $j = i_t$ at each round t, the selected algorithm a^{i_t} updates its learning state, while with the remaining probability, its feedback gets blocked. The selection of this random variable ensures that the feedback of each algorithm is allowed, on average, with constant probability $\rho = \eta/A$ over the whole horizon T. The analytical steps of this learning scheme are shown in Algorithm 2.

It is crucial to stress that the regret of the meta-learner w.r.t. the best algorithm, cf. (17), is uninformative on its own in the bandit setting. The reason can be attributed to the indirect association between rewards at any given time t and the algorithms the meta-learner previously selected. The past

[9]This is a natural approach for our problem setting, as each algorithm proposes possibly different policies at each round, but only the policy of one algorithm can be deployed to the vBS and return a reward.

[10]We recall that no assumptions are made about the sequence of rewards $\{f_t\}_{t=1}^T$, which can even be chosen from an adversary, as described analytically in Sec. III.

selections define the current learning state of the algorithms, which, in turn, impacts the rewards [41]. Therefore, the evaluation should contain a comparison to an ideal policy that consistently selects the best algorithm, which obtains feedback in every t and performs well with respect to the single best policy. Formally, we are interested in minimizing the regret of the meta-learner w.r.t. the single best policy, which is equal to:

$$\mathcal{R}_T^{\mathrm{M}} = \underbrace{\max_{x\in\mathcal{X}^{i^*}} \left\{\sum_{t=1}^{T} f_t(x)\right\}}_{\text{best policy}} - \underbrace{\mathbb{E}\left[\sum_{t=1}^{T} f_t\big(a^{i_t}(h_t^{i_t,\mathrm{S}})\big)\right]}_{\text{meta-learner}}. \quad (14)$$

The aggregate reward of the best algorithm a^{i^*} achieved until round t is:

$$\max_{j\in\mathcal{A}} \left\{\sum_{t=1}^{T} \mathbb{E}\left[f_t\big(a^{j}(h_t^{j,\mathrm{S}})\big)\right]\right\} \equiv \sum_{t=1}^{T} \mathbb{E}\left[f_t\big(a^{i^*}(h_t^{i^*,\mathrm{S}})\big)\right]. \quad (15)$$

We add and subtract (15) from (14), and we derive:

$$\mathcal{R}_T^{\mathrm{M}} = \mathcal{R}_T^{\mathrm{M}_1} + \mathcal{R}_T^{\mathrm{M}_2}, \quad (16)$$

where $\mathcal{R}_T^{\mathrm{M}_1}$ corresponds to the regret of the meta-learner with respect to the best algorithm:

$$\mathcal{R}_T^{\mathrm{M}_1} = \underbrace{\sum_{t=1}^{T} \mathbb{E}\left[f_t\big(a^{i^*}(h_t^{i^*,\mathrm{S}})\big)\right]}_{\text{best algorithm}} - \underbrace{\mathbb{E}\left[\sum_{t=1}^{T} f_t\big(a^{i_t}(h_t^{i_t,\mathrm{S}})\big)\right]}_{\text{meta-learner}}, \quad (17)$$

and $\mathcal{R}_T^{\mathrm{M}_2}$ corresponds to the regret of the best algorithm w.r.t. to the best policy:

$$\mathcal{R}_T^{\mathrm{M}_2} = \underbrace{\max_{x\in\mathcal{X}^{i^*}} \left\{\sum_{t=1}^{T} f_t(x)\right\}}_{\text{best policy}} - \underbrace{\sum_{t=1}^{T} \mathbb{E}\left[f_t\big(a^{i^*}(h_t^{i^*,\mathrm{S}})\big)\right]}_{\text{best algorithm}}. \quad (18)$$

If a^{i^*} had access to its full history $h_t^{i^*}$, we denote as $\beta^{i^*} \in [0,1]$ the exponent of the upper bound of its regret, namely[11]:

$$\max_{x\in\mathcal{X}^{i^*}} \left\{\sum_{t=1}^{T} f_t(x)\right\} - \sum_{t=1}^{T} \mathbb{E}\left[f_t\big(a^{i^*}(h_t^{i^*})\big)\right] \leq \mathcal{O}(T^{\beta^{i^*}}).$$

However, in the considered analysis, it has access to its partial history $h_t^{i^*,\mathrm{S}}$. For proving a non-trivial upper bound on $\mathcal{R}_T^{\mathrm{M}}$ in this case, the best performing algorithm a^{i^*} should satisfy the following:

$$\max_{x\in\mathcal{X}^{i^*}} \left\{\sum_{t=1}^{T} f_t(x)\right\} - \sum_{t=1}^{T} \mathbb{E}\left[f_t\big(a^{i^*}(h_t^{i^*,\mathrm{S}})\big)\right] \leq \mathcal{O}\left(\frac{(\rho T)^{\beta^{i^*}}}{\rho}\right), \quad (19)$$

where $\rho = \eta/A$, as defined beforehand. A rich class of online learning algorithms, including Exp3 (and thus, BSvBS), satisfy (19), which, in turn, quantifies the robustness of an online learning algorithm w.r.t. the sparsity of the history [40].

[11]For instance, if BSvBS is the best algorithm a^{i^*}, then $\beta^{i^*} = 1/2$, see Lemma 1.

C. Optimality Guarantees

The performance of Algorithm 2 is captured by the following lemma:

Lemma 2: Let $T > 0$ be a fixed time horizon, and assume the best algorithm, a^{i^*}, satisfies (19) with β^{i^*}. Set input parameter $\eta = \Theta\big(T^{-\frac{1-\beta}{2-\beta}} A^{\frac{1-\beta}{2-\beta}} (\log A)^{\frac{1}{2}\mathbb{1}_{\{\beta=0\}}}\big)$, where $\beta \geq \beta^{i^*}$. Then, running Algorithm 2 ensures that the expected regret is sub-linear:

$$\mathcal{R}_T^{M} \leq \mathcal{O}\big(T^{\frac{1}{2-\beta}} A^{\frac{1}{2-\beta}} (\log A)^{\frac{1}{2}\mathbb{1}_{\{\beta=0\}}}\big) \quad (20)$$

Proof: The proof follows by tailoring the main result of [40]; we therefore provide a brief but sufficient explanation. By applying Lemma 1, (17) gives:

$$\mathcal{R}_T^{\mathrm{M}_1} \leq c\,\eta\,T + \frac{A\log A}{\eta}, \quad (21)$$

where $c > 0$ is a constant. Adding (19) and (21), results in:

$$\mathcal{R}_T^{\mathrm{M}} \leq \mathcal{O}\big(\eta\,T + \frac{A\log A}{\eta} + \frac{T^{\beta^{i^*}} A^{1-\beta^{i^*}}}{\eta^{1-\beta^{i^*}}}\big). \quad (22)$$

Setting $\eta \sim T^{-z}$ and finding the z that minimizes the power of T in (22), leads to (20). □

D. Discussion for MetBS

When interacting with *learning algorithms* in the *bandit* setting, Algorithm 2 is guaranteed to achieve the same performance as the best algorithm if it ran on its own (and thus, acquiring feedback in every round). Hence, MetBS attains reward as the (unknown) single best algorithm without making assumptions for the environment (see Lemma 2). This accomplishment is made possible through minimum coordination between the meta-learner and the algorithms, as described in lines 10-11 of Algorithm 2.

In terms of implementation, MetBS can be implemented as another rApp, which also facilitates its coordination with the co-located rApps implementing the different algorithms; see also Fig. 1(c). Regarding its overheads, due to its similarity with BSvBS, its complexity depends on the number of algorithms that it chooses from, i.e., $\mathcal{O}(T|\mathcal{A}|)$ for T rounds. However, as it chooses between different algorithms (where each of them selects policies and has its own complexity), the overall time complexity of MetBS depends on the worst-case scenario of the most time-complex algorithm. Similarly, its space complexity is equal to $\mathcal{O}(|\mathcal{A}|)$; however, an important factor is the complexity of the algorithms that it chooses from, and especially, the most space-complex algorithm.

VI. Performance Evaluation

A. Experimental Setup & Scenarios

The solutions are assessed under different traffic and environment scenarios using our recent publicly-available dataset [7] with power consumption and throughput measurements from an O-RAN compatible testbed. This experimental

setup includes a vBS and a UE,[12] implemented as srseNB and srsUE from the srsRAN suite [2]. The RUs of the vBS and UE are composed of an Ettus Research USRP B210, and their BBUs and near-RT RICs are implemented on general-purpose computers (Intel NUC BOXNUC8I7BEH). The power consumption of the BBU and RU is measured with the GW-Instek GPM-8213. A 10 MHz band is selected, supplying a maximum capacity of approximately 32 Mbps and 23 Mbps for the downlink and uplink operation, respectively. The non-RT threshold policies are calculated in a programming language, emulating the operation of rApps; the real-time scheduling decisions are made by the default srsRAN scheduler that has been amended to comply with the MCS, PRB, and power thresholds that are provided to them in each round.

The dataset contains 32 797 measurements for $|\mathcal{X}| = 1080$ policies corresponding to $\mathcal{B}^{\mathrm{d}} = \{0, 0.2, 0.6, 0.8, 1\}$, $\mathcal{B}^{\mathrm{u}} = \{0.01, 0.2, 0.4, 0.6, 0.8, 1\}$, $\mathcal{M}^{\mathrm{d}} = \{0, 5, 11, 16, 22, 27\}$, $\mathcal{P}^{\mathrm{d}} = \{3\}$[13] and $\mathcal{M}^{\mathrm{u}} = \{0, 5, 9, 14, 18, 23\}$. The random perturbations in this setup, as explained in Sec. III, emanate due to time-varying UL and DL demands, $\{d_t^{\mathrm{u}}, d_t^{\mathrm{d}}\}_{t=1}^{T}$, measured in Mbps, and time-varying CQIs, $\{c_t^{\mathrm{u}}, c_t^{\mathrm{d}}\}_{t=1}^{T}$, which are dimensionless. The transmitted data, $\{R_t^{\mathrm{u}}, R_t^{\mathrm{d}}\}_{t=1}^{T}$, are calculated by multiplying the values of $\mathcal{B}^{\mathrm{d}}$ ($\mathcal{B}^{\mathrm{u}}$) with the transport block size (TBS); the latter is determined by mapping the $\mathcal{M}^{\mathrm{d}}$ ($\mathcal{M}^{\mathrm{u}}$) with the TBS index [42]. W.l.o.g., we have assumed 50 PRBs. The power cost function is set to $P_t(x_t) = V_t$, where V_t is the total power consumed by the vBS, and the utility function as stated in (2). W.l.o.g., we scale both components of the reward function to $[0, 1]$ and choose $\delta = 1.5$ to prioritize the power consumption unless stated otherwise. We set $\gamma = 0.29$ for `BSvBS` and $\eta = 0.04$ for `MetBS`, and use $T = 50k$. All results are averaged over 10 independent experiments.

For the ensuing analysis, we assess three scenarios which represent a static environment (fixed, time-invariant parameters); a stationary stochastic environment (i.i.d. parameters); and an adversarial scenario. The latter, clearly, is an extreme case (e.g., can appear under high mobility conditions, heavy interference or attacks) that we use to demonstrate the robustness of the learning algorithms. On the other hand, the first two scenarios are in line with those typically considered by prior benchmarks, e.g., [7] and [11]. In detail:

- ***Scenario A (static)***: the demands and CQIs take the highest possible values according to our testbed, i.e., $d_t^{\mathrm{d}} = 32$, $d_t^{\mathrm{u}} = 23$, $c_t^{\mathrm{d}} = 15$, $c_t^{\mathrm{u}} = 15$.
- ***Scenario B (stationary)***: the demands and CQIs are drawn randomly from fixed uniform distributions in each round, i.e., $d_t^{\mathrm{d}} \sim \mathcal{U}(29, 32)$, $d_t^{\mathrm{u}} \sim \mathcal{U}(20, 23)$, $c_t^{\mathrm{d}}, c_t^{\mathrm{u}} \sim \mathcal{U}(1, 3)$, where $\mathcal{U}(a, b)$ denotes the uniform distribution over the interval $[a, b]$.
- ***Scenario C (adversarial)***: the demands and CQIs are drawn randomly in a *ping-pong* way; namely, in *odd* rounds according to $d_t^{\mathrm{d}} \sim \mathcal{U}(29, 32)$, $d_t^{\mathrm{u}} \sim \mathcal{U}(20, 23)$, $c_t^{\mathrm{d}}, c_t^{\mathrm{u}} \sim \mathcal{U}(13, 15)$, and in *even* rounds from $d_t^{\mathrm{d}}, d_t^{\mathrm{u}} \sim \mathcal{U}(0.01, 1)$, $c_t^{\mathrm{d}}, c_t^{\mathrm{u}} \sim \mathcal{U}(1, 3)$.[14] We note note that the learner does not have access to this information, and is oblivious to when the switches happen.

[12]The usage of one UE is not limiting for our study, since the algorithm devises each vBS's thresholds based on the average (across users) throughput and energy, and the average CQI and traffic, i.e., the UE emulates the load of multiple users.

[13]The DL transmission power is determined through the transmission gain of the USRP implementing the BS. The RU of the testbed is equipped with a fixed power amplifier that consumes 3 W and a variable attenuator for power calibration. To account for this limitation, the dataset power measurements are post-processed using linear modeling [7].

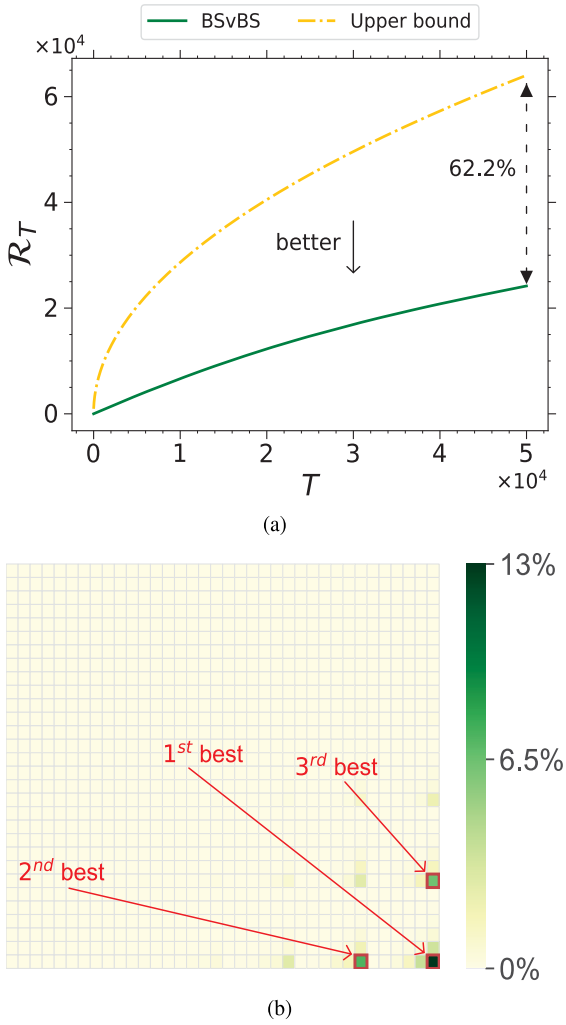

Fig. 2. **(a)** R_T achieved from `BSvBS` in Scenario A (static) and its upper bound; **(b)** heatmap for the choices of `BSvBS` in Scenario A, showing the probability that each policy is chosen at $t = 50k$.

Scenario C resembles dynamic environments, where the parameters might change drastically every round. This corresponds to the most challenging-to-learn *adversarial* schemes in regret analysis, cf. [25]. Clearly, an algorithm that performs well under this case is guaranteed to perform well in all other scenarios. In the sequel, we use these scenarios to explore the convergence of the proposed learning and meta-learning algorithms, and compare them with selected state-of-the-art competitors in terms of *(i)* regret, *(ii)* vBS power savings, and *(iii)* inference time.

B. Static & Stationary Scenarios

Fig. 2(a) shows the expected regret in Scenario A when prioritizing the utility function (small δ). The attained regret is

[14]CQI 13 and 15 correspond to SNR of 25 dB and 29 dB, while CQI 1 and 3 to SNR of 1.95 dB and 6 dB, respectively.

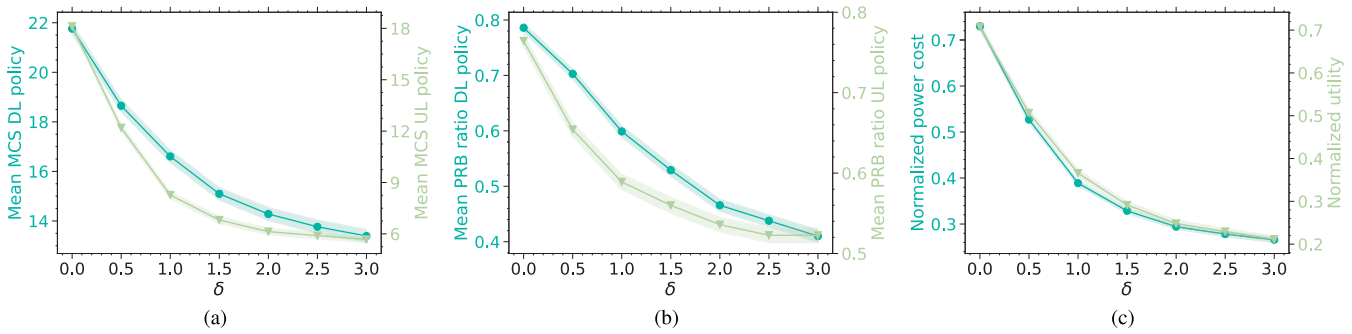

Fig. 3. Scenario A (static) for BSvBS: **(a)** MCS in DL (left) / UL (right); **(b)** PRB ratio in DL (left) / UL (right); **(c)** power (left) and utility (right) w.r.t. δ, with 0.95-CI. In each plot, the blue and green lines correspond to the left and right y-axis, respectively.

sub-linear and $62.2\,\%$ smaller than the upper bound (which is itself sub-linear), cf. (8). To complement the analysis, Fig. 2(b) shows a grid with 1080 cells, each mapping a different policy. The cells are colored based on the probability BSvBS selects each policy at $t = 50k$, where darker colors indicate higher probabilities. The red squares indicate the three best policies chosen $25\,\%$ of the rounds, where the top-performing one is selected twice as frequently. This outcome can be attributed to the small δ, which favors the policy with the highest MCS and PRB ratio in both DL and UL, as the demands and CQIs are high. For the second and third-best policies, the MCS in UL and DL take the highest values, except for the PRB ratios, which are fixed at 0.8, namely, the second-best UL and DL PRB ratios.

Fig. 3(a) and 3(b) delineate the effect of δ on the MCS DL/UL, and PRB ratio DL/UL, respectively (i.e., the chosen policies), for the static scenario. The solid lines in the plots represent the mean values averaging 100 rounds after running BSvBS for $t = 50k$ rounds, and the shadowed areas are the 0.95-confidence intervals. Moreover, the blue and green lines correspond to the left and right y-axis, respectively. We observe that smaller δ leads to higher MCS and PRB ratio choices in DL and UL. This is justified by the high CQI values considered in this scenario, as they enable using higher MCS, which allows more data transmission and larger decoding computational load [43]. Furthermore, larger δ in Scenario A effectuates the selection of lower MCS and PRB values in order for the vBS to save resources by diminishing the turbo decoding iterations.

Similarly, Fig. 3(c) illustrates the impact of δ on the reward function, where its two components are normalized, see (2). Higher δ boosts the usage of policies that minimize the consumed power, forcing the utility function to decrease, whereas lower δ leads to policies that maximize the utility but increase the power consumption. Values $\delta > 2$ have less effect on the power and utility functions, as there is a limit in the consumed power that can be saved.

Fig. 4 depicts the average regret over time for stationary Scenario B, which converges towards zero as time elapses. We also plot the average regret of a typical benchmark that randomly selects policies with equal probability; we call this benchmark Random. BSvBS explores policies with probability $29\,\%$ (since $\gamma = 0.29$) and exploits the best-performing ones with probability $71\,\%$. Therefore, in the first 800 rounds,

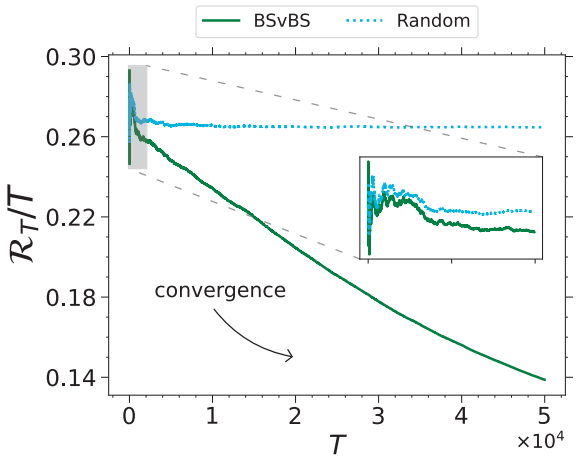

Fig. 4. R_T/T for BSvBS in Scenario B (stationary), together with Random, a naive algorithm that selects policies randomly.

BSvBS obtains similar regret as the benchmark algorithm, but their performance difference grows gradually, reaching $33.3\,\%$ in round $t = 50k$, as BSvBS opts for the best-performing policies with higher probability at latter stages.

Key takeaways: *(i)* The measured regret is sub-linear in static and stationary scenarios and substantially smaller (up to $62.2\,\%$) than the theoretical bound. *(ii)* The network can adjust δ to trade certain power consumption with commensurate losses in utility; yet, increasing δ more than a specific value ($\delta = 2$ in our case) does not provide further substantial savings.

C. Gap in Prior Work

The primary objective is to showcase how state-of-the-art techniques perform inadequately in challenging environments. To delineate this effect, we focus on a smaller set of policies, i.e., $|\mathcal{M}_d| = |\mathcal{M}_u| = |\mathcal{B}_u| = |\mathcal{B}_d| = 2$ and $|\mathcal{P}_d| = 1$, yielding $|\mathcal{X}| = 16$ policies. The performance of the BP-vRAN algorithm [7], which constitutes, to the best of the authors' knowledge, the only existing work designed to configure such threshold policies in vBS, is assessed in adversarial Scenario C. BP-vRAN, which is based on the seminal GP-UCB algorithm [44], models the traffic demands and CQIs as *context*, which are observed before the policy is decided. Given that the context directly impacts the selection of policies, it will be shown how abrupt changes in CQI values and traffic

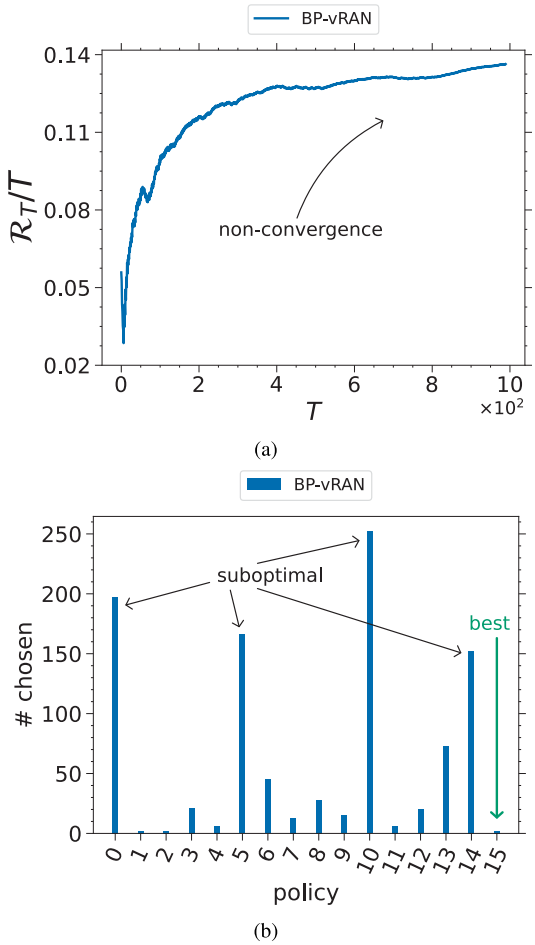

Fig. 5. BP−vRAN executed for $T = 1000$ rounds in dynamic Scenario C, in a subset of the policy space: **(a)** R_T/T; **(b)** number of times each policy is chosen.

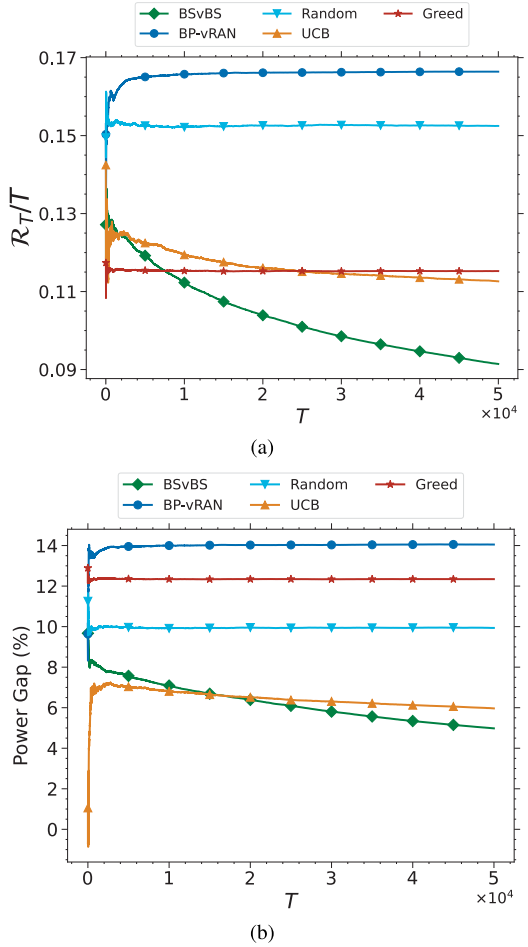

Fig. 6. Comparison of BSvBS with several competitors in adversarial Scenario C: **(a)** R_T/T; **(b)** power saving of each algorithm with respect to the ideal-minimum energy of the benchmark.

demand deteriorate the algorithm performance. We present an example where the context differs between its observation and application to the system. This case appears quite often in practice, given that the rounds of reference are of several seconds. For the plots in this section, the reward function $f_t(x_t)$ is unbounded.[15]

As indicated in Fig. 5(a), the average regret in the adversarial Scenario C does not decrease (in fact, it increases) after $T = 1k$ rounds, which is more than 33× of the advertised convergence time. This happens because the algorithm takes decisions in each t by assuming perfect knowledge of f_t, which might take arbitrary values depending on the environment. Clearly, due to the system's volatility, the policy for each t should be selected based on past values $\{f_\tau(x_\tau)\}_{\tau=1}^{t-1}$; yet, as Fig. 5(b) corroborates, BP−vRAN selects sub-optimal policies for most rounds and fails to explore efficiently even this small space.

D. Evaluation of the Bandit Algorithm

Fig. 6(a) compares the average regret over time of BSvBS for Scenario C, in relation to several competitor algorithms, namely: the BP−vRAN; a naive algorithm that selects thresholds uniformly randomly (Random); the classical UCB algorithm that is designed for stationary environments [45]; and a greedy algorithm that prioritizes exploitation (Greed, selects the best solution found until now) [46]. We consider $T = 50k$ rounds and use the complete policy space (i.e., $|\mathcal{X}| = 1080$), and all results are averaged over 10 independent

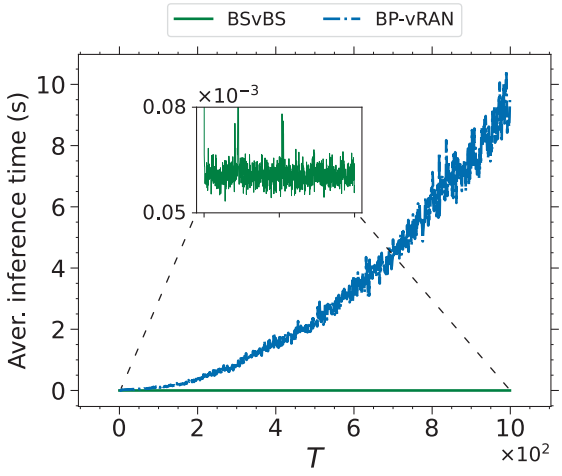

Fig. 7. Average time needed to infer a policy in each round, for our algorithm BSvBS, and its main competitor, BP−vRAN.

[15]When BSvBS is depicted in the same plot as BP−vRAN, the reward function of BP−vRAN is scaled too.

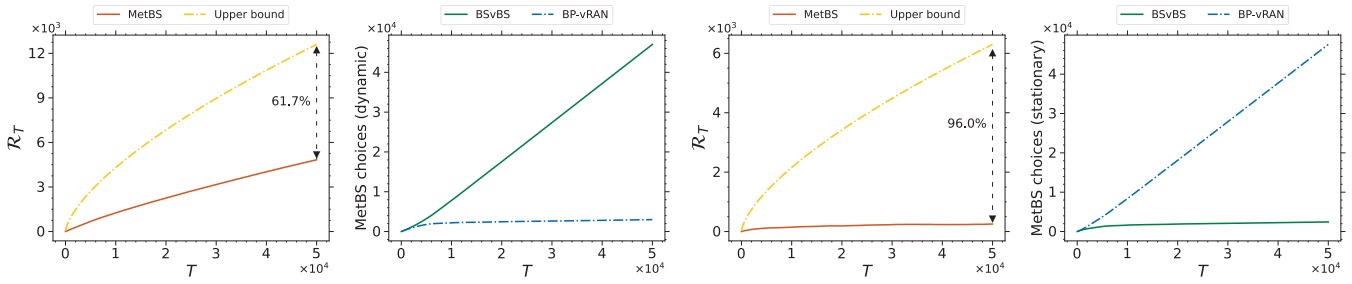

Fig. 8. Meta-learning algorithm: R_T and the upper bound for dynamic **(a)** and stationary **(c)** scenarios; number of times BSvBS and BP−vRAN are chosen in $T = 50k$ rounds for dynamic **(b)** and stationary **(d)** scenarios.

experiments. We observe that BSvBS is superior, acquiring 45.1% less regret w.r.t BP−vRAN, and 22% less w.r.t Greed and UCB at $t = 50k$. It is worth noting that Random performs better than BP−vRAN in this case, by approximately 9%.

In Fig. 6(b), we present the vBS power gains that each algorithm achieves in the same scenario, w.r.t. the ideal-minimum-energy of the benchmark, where the power consumption of the idle user is subtracted. It can be seen that with BSvBS, the network operator can save up to 64.5% of energy if the algorithm runs for $t = 50k$ rounds in contrast to BP−vRAN. Moreover, it can be seen that UCB also chooses policies that allow for saving energy, but again, attains less energy saving than BSvBS. These plots also showcase that the Greed algorithm, which does not explore new policies, is not competitive and is stuck in exploiting sub-optimal policies (straight line in the regret plot).

Another key advantage of BSvBS is its low inference time, i.e., the time to deduce a policy in each round. Fig. 7 exhibits the average inference time and compares it with BP−vRAN. Using standard kernel-based methods (as BP−vRAN does) is widely recognized to result in a high computational cost of $\mathcal{O}(t^3)$ with respect to the number of data points t [47]. This is a significant limitation as it delays the vBS operation to more than 10 s after $t = 1k$ when tested on an Apple M1 chip with 8-core CPU@3.2 GHz. Clearly, this hinders the vBS operation, which will then have to rely on stale information. On the other hand, we notice that BSvBS requires no more than 0.08 ms to decide a policy, which remains constant throughout.

Key takeaways: In challenging (i.e., non-stationary / adversarial) environments, decisions for configuring the vBS should be taken based on past performance. Requiring *perfect* knowledge of the environment could lead to sub-optimal policies, increasing power costs up to 64.5 % for operators. BSvBS's performance is robust to such adversarial scenarios and outperforms a state-of-the-art algorithm in terms of: *(i)* the average regret (up to 45.1 % superiority), *(ii)* the power gap w.r.t. the minimum vBS energy consumption (up to 64.5 % superiority), and *(iii)* inference time (solely 0.08 ms). We recall that BSvBS does not have access to how and when the demands and CQI change.

E. Evaluation of the Meta-Learning Algorithm

We consider $A = 2$ with BP−vRAN, and BSvBS that select policies from $\mathcal{X}$. On the one hand, if the context is not available at the beginning of each round, as happens in several real-world applications, BSvBS is superior and BP−vRAN fails, as seen in Sec. VI. Hence, MetBS opts mainly for BSvBS. The attained regret (Fig. 8(a)) is by 61.7 % less than the upper bound, which implies the desired sub-linear regret. The algorithms that MetBS chooses can be verified in Fig. 8(b), where BSvBS is selected in approximately $47k$ rounds, while the sub-optimal BP−vRAN in the remaining $3k$ rounds ($T = 50k$). On the other hand, if the environment is easy, BP−vRAN is expected to converge faster than BSvBS;

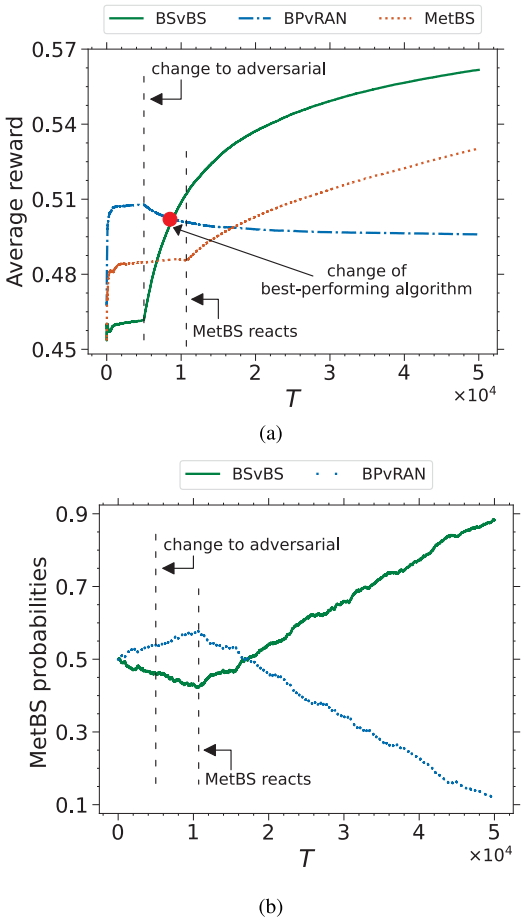

Fig. 9. **(a)** Average reward and **(b)** probabilities that BSvBS and BP−vRAN are chosen by the MetBS when the environment changes (leftmost dashed line, *change to adversarial*) from stationary to adversarial at $t = 5k$. The red dot (*change of best-performing algorithm*) shows the change of the best-performing algorithm (from BP−vRAN to BSvBS), and the rightmost dashed line MetBS reacts) depicts the round after which MetBS starts choosing the best-performing algorithm, BSvBS, more often.

and, as a consequence, to be preferred by the meta-learner. Indeed, the regret of MetBS (Fig. 8(c)) is 96 % lower than the upper bound stated in (8), which clearly indicates the expected sub-linear regret has been achieved. MetBS selects BP−vRAN in roughly $46k$ rounds, while BSvBS in $4k$ rounds (Fig. 8(d)). It is important to heed that BSvBS converges as well to the optimal policy but slower (see Fig. 2 and Fig. 4), an unavoidable side-effect of its robustness under any environment (even adversarial).

Finally, we test the meta-learner in a "mixed" environment, where, in the first $5k$ rounds the demands and CQIs are drawn from Scenario B (stationary), and in the remaining $45k$ rounds from Scenario C (adversarial). Fig. 9(a) depicts the average rewards of MetBS, BSvBS, and BP−vRAN. It can be viewed that before the change of the environment, the average reward of the meta-learner follows closer to the reward of BP−vRAN; the orange dotted line is 3.8 % lower than the blue dash-dotted line. The same can be verified from Fig. 9(b), where BP−vRAN is chosen with higher probability, 58 %, before $t = 5k$. When the change occurs, MetBS does not opt immediately for BSvBS, as the average reward of BP−vRAN is still higher, until the change-point at roughly $t = 8k$, which is shown with a red dot in Fig. 9(a). After this round, BSvBS experiences larger reward values on average, and within less than $1k$ rounds (i.e., 2 %), MetBS starts indeed selecting BSvBS more frequently (up to 88.2 %).

Key takeaways: MetBS chooses the best-performing algorithm for each scenario. When the demands and CQIs are drawn from a stationary distribution, it prioritizes BP−vRAN (92 % of rounds), while in adversarial scenarios, it follows BSvBS (94 % of rounds). In mixed scenarios, MetBS tracks and applies the changes after only 2 % of rounds.

VII. Conclusion and Future Work

The virtualization of base stations and the design of O-RAN systems are instrumental for the success of the next generation of mobile networks. Allocating resources for these vBSs by choosing policies that operate on a longer time scale and do not require intervention on the (often proprietary) vBS node implementations is a new and promising network control approach. However, in order to be practical and successful, the proposed solutions should have low overhead and require no assumption about the future channel qualities and traffic demands (i.e., the environment).

The first proposed scheme possesses exactly these properties, building on a tailored adversarial learning algorithm that has minimal overhead and can run in sub-milliseconds. In line with prior works, we focus on the important metrics of throughput and energy consumption and explore their trade-offs in a range of scenarios with experimental datasets. As this robustness comes at a cost for convergence speed, we aim to increase the latter in easy scenarios, where the environment is known beforehand (or changes slowly), through a meta-learning scheme that combines a mix of algorithms, including our own, and delineates the best-performing one at runtime. This creates a best-of-both-worlds solution. Our extensive data-driven experiments demonstrate energy savings up to 64.5 % compared to state-of-the-art competitors.

We highlight that the regret depends on the number of possible policies up to a square-root factor. While their number is expected to be smaller than the number of policies applied to the RT O-RAN level, this finding still points to an interesting direction for further reducing this dependency. Finally, exploring the interactions between rApps and xApps in a real-time setting and assessing their impact on network performance is an interesting direction for future work, by expanding, e.g., [48].

References

[1] A. Garcia-Saavedra and X. Costa-Perez, "O-RAN: Disrupting the virtualized RAN ecosystem," *IEEE Commun. Standards Mag.*, vol. 5, no. 4, pp. 96–103, Oct. 2021.

[2] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "SrsLTE: An open-source platform for LTE evolution and experimentation," in *Proc. ACM WinTECH*, Oct. 2016, pp. 25–32.

[3] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "Openairinterface: A flexible platform for 5G research," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 33–38, 2014.

[4] A. Alnoman, G. H. S. Carvalho, A. Anpalagan, and I. Woungang, "Energy efficiency on fully cloudified mobile networks: Survey, challenges, and open issues," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1271–1291, 2nd Quart., 2018.

[5] P. Rost, A. Maeder, M. C. Valenti, and S. Talarico, "Computationally aware sum-rate optimal scheduling for centralized radio access networks," in *Proc. IEEE GLOBECOM*, Dec. 2015, pp. 1–6.

[6] J. A. Ayala-Romero, I. Khalid, A. Garcia-Saavedra, X. Costa-Perez, and G. Iosifidis, "Experimental evaluation of power consumption in virtualized base stations," in *Proc. ICC*, Jun. 2021, pp. 1–6.

[7] J. A. Ayala-Romero, A. Garcia-Saavedra, X. Costa-Perez, and G. Iosifidis, "Bayesian online learning for energy-aware resource orchestration in virtualized rans," in *Proc. IEEE INFOCOM*, May 2021, pp. 1–10.

[8] O-RAN Alliance, *O-RAN Architecture Description 12.0 (O-RAN.WG1.OAD-R003-v12.00)*, Tech. Specification, Alfter, Germany, Jun. 2024.

[9] O-RAN Alliance, *O-RAN Cloud Architecture and Deployment Scenarios for O-RAN vRAN 7.0 (O-RAN.WG6.CADS-v07.00)*, Tech. Rep., Alfter, Germany, Jun. 2024.

[10] J. A. Ayala-Romero, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Perez, A. Banchs, and J. J. Alcaraz, "VrAIn: A deep learning approach tailoring computing and radio resources in virtualized RANs," in *Proc. ACM Mobicom*, Oct. 2019, pp. 1–16.

[11] J. A. Ayala-Romero, A. Garcia-Saavedra, X. Costa-Perez, and G. Iosifidis, "EdgeBOL: Automating energy-savings for mobile edge AI," in *Proc. ACM CoNEXT*, Dec. 2021, pp. 397–410.

[12] M. Kalntis and G. Iosifidis, "Energy-aware scheduling of virtualized base stations in O-RAN with online learning," in *Proc. GLOBECOM*, Dec. 2022, pp. 6048–6054.

[13] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, and Z. Smoreda, "Identifying common periodicities in mobile service demands with spectral analysis," in *Proc. MedComNet*, Jun. 2020, pp. 1–8.

[14] D. Bega, A. Banchs, M. Gramaglia, X. Costa-Pérez, and P. Rost, "CARES: Computation-aware scheduling in virtualized radio access networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 12, pp. 7993–8006, Dec. 2018.

[15] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2224–2287, 3rd Quart., 2019.

[16] A. Zappone, M. Di Renzo, and M. Debbah, "Wireless networks design in the era of deep learning: Model-based, AI-based, or both?" *IEEE Trans. Commun.*, vol. 67, no. 10, pp. 7331–7376, Oct. 2019.

[17] J. J. Alcaraz, J. A. Ayala-Romero, J. Vales-Alonso, and F. Losilla-López, "Online reinforcement learning for adaptive interference coordination," *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 10, Oct. 2020, Art. no. e4087.

[18] F. W. Murti, J. A. Ayala-Romero, A. Garcia-Saavedra, X. Costa-Pérez, and G. Iosifidis, "An optimal deployment framework for multi-cloud virtualized radio access networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2251–2265, Apr. 2021.

[19] Y. Cao, S.-Y. Lien, Y.-C. Liang, K.-C. Chen, and X. Shen, "User access control in open radio access networks: A federated deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 21, no. 6, pp. 3721–3736, Jun. 2022.

[20] A. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," in *Discrete Event Dynamic Systems: Theory and Applications*, vol. 13, 2002.

[21] B. Alt, T. Ballard, R. Steinmetz, H. Koeppl, and A. Rizk, "CBA: Contextual quality adaptation for adaptive bitrate video streaming," in *Proc. IEEE INFOCOM*, Apr. 2019, pp. 1000–1008.

[22] J. Chuai et al., "A collaborative learning based approach for parameter configuration of cellular networks," in *Proc. IEEE INFOCOM*, Apr. 2019, pp. 1396–1404.

[23] M. A. Qureshi and C. Tekin, "Fast learning for dynamic resource allocation in AI-enabled radio networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 1, pp. 95–110, Mar. 2020.

[24] L. Maggi, A. Valcarce, and J. Hoydis, "Bayesian optimization for radio resource management: Open loop power control," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 1858–1871, Jul. 2021.

[25] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," 2012, *arXiv:1204.5721*.

[26] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, Jan. 2016.

[27] A. Careem, M. Ahamed, and A. Dutta, "Real-time prediction of non-stationary wireless channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 7836–7850, Dec. 2020.

[28] G. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, "Wireless caching: Technical misconceptions and business barriers," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 16–22, Aug. 2016.

[29] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Inf. Comput.*, vol. 108, no. 2, pp. 212–261, Feb. 1994.

[30] A. Sani, G. Neu, and A. Lazaric, "Exploiting easy data in online optimization," in *Proc. NeurIPS*, 2014, pp. 810–818.

[31] S. Zarandi, A. Khalili, M. Rasti, and H. Tabassum, "Multi-objective energy efficient resource allocation and user association for in-band full duplex small-cells," *IEEE Trans. Green Commun. Netw.*, vol. 4, no. 4, pp. 1048–1060, Dec. 2020.

[32] Y. Wu, F. Zhou, W. Wu, Q. Wu, R. Q. Hu, and K.-K. Wong, "Multi-objective optimization for spectrum and energy efficiency tradeoff in IRS-assisted CRNs with NOMA," *IEEE Trans. Wireless Commun.*, vol. 21, no. 8, pp. 6627–6642, Aug. 2022.

[33] X. Qi, S. Khattak, A. Zaib, and I. Khan, "Energy efficient resource allocation for 5G heterogeneous networks using genetic algorithm," *IEEE Access*, vol. 9, pp. 160510–160520, 2021.

[34] F. Meng, P. Chen, L. Wu, and J. Cheng, "Power allocation in multi-user cellular networks: Deep reinforcement learning approaches," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6255–6267, Oct. 2020.

[35] M. Tsampazi et al., "A comparative analysis of deep reinforcement learning-based xApps in O-RAN," in *Proc. GLOBECOM*, Dec. 2023, pp. 1638–1643.

[36] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "ColO-RAN: Developing machine learning-based xApps for open RAN closed-loop control on programmable experimental platforms," *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 5787–5800, Oct. 2022.

[37] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The non-stochastic multiarmed bandit problem," *SIAM J. Comput.*, vol. 32, no. 1, pp. 48–77, 2002.

[38] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[39] A. Agarwal, H. Luo, B. Neyshabur, and R. E. Schapire, "Corralling a band of bandit algorithms," in *Proc. COLT*, 2017, pp. 12–38.

[40] A. Singla, H. Hassani, and A. Krause, "Learning to interact with learning agents," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 4083–4090.

[41] M. Odalric and R. Munos, "Adaptive bandits: Towards the best history-dependent strategy," in *Proc. AISTATS*, 2011, pp. 570–578.

[42] *Physical Layer Procedures*, document TS 36.213, 3rd Generation Partnership Project (3GPP), 2024.

[43] P. Rost, S. Talarico, and M. C. Valenti, "The complexity–rate tradeoff of centralized radio access networks," *IEEE Trans. Wireless Commun.*, vol. 14, no. 11, pp. 6164–6176, Nov. 2015.

[44] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proc. ICML*, 2010, pp. 1015–1022.

[45] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2, pp. 235–256, 2002.

[46] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., Cambridge, MA, USA: MIT Press, 2018.

[47] S. Vakili, J. Scarlett, D.-S. Shiu, and A. Bernacchia, "Improved convergence rates for sparse approximation methods in kernel-based learning," in *Proc. ICML*, 2022, pp. 21960–21983.

[48] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "OpenRAN gym: An open toolbox for data collection and experimentation with AI in O-RAN," in *Proc. IEEE WCNC*, Apr. 2022, pp. 518–523.

Michail Kalntis (Graduate Student Member, IEEE) received the Diploma degree in electrical and computer engineering and the M.Sc. degree in computer science from the National Technical University of Athens, Greece, in 2019 and 2021, respectively. During his studies, he was a recipient of the Merit Scholarship from Propondis Foundation. He is currently pursuing the Ph.D. degree with the Networked Systems Group, Delft University of Technology, The Netherlands. His research interests lie in mobile networks and network optimization.

George Iosifidis (Member, IEEE) received the Diploma degree in electronics and telecommunications engineering from the Greek Air Force Academy, Athens, in 2000, and the Ph.D. degree from the University of Thessaly in 2012. He was an Assistant Professor with the Trinity College Dublin from 2016 to 2020 and a Research Scientist with Yale University (2015–2017). He is currently an Associate Professor with Delft University of Technology. His research interests lie in the broad area of network optimization and economics. For more information visit the link (https://www.futurenetworkslab.net/).

Fernando A. Kuipers (Senior Member, IEEE) is currently a Full Professor with Delft University of Technology (TU Delft), where he established and leads the Networked Systems Group and the Laboratory on Internet Science. He was a Visiting Scholar with Technion, Israel Institute of Technology, in 2009, and Columbia University, New York City, in 2016. His research comprises network optimization, network resilience, quality-of-service, quality-of-experience and addresses problems in computer networks, software-defined networking, 6G, and the Internet of Things. His work on these subjects led to a Ph.D. degree conferred cum laude in 2004, the highest possible distinction with TU Delft, and includes distinguished papers at IEEE INFOCOM 2003, Chinacom 2006, IFIP Networking 2008, IEEE FMN 2008, IEEE ISM 2008, ITC 2009, IEEE JISIC 2014, NetGames 2015, and EuroGP 2017. He has served as the General Chair and the TPC Chair in flagship conferences, such as ACM SIGCOMM (2021 and 2022) and IEEE INFOCOM (2024) and the Vice Chair of the ACM SIGCOMM Executive Committee. He co-founded the Do IoT fieldlab and the PowerWeb Institute and served on the board for the TU Delft Safety & Security Institute. Currently, he is a Co-PI of the Dutch 6G Flagship Project Future Network Services, where he leads the program line intelligent networks.